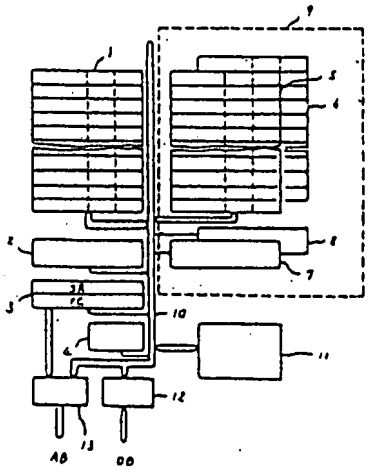
(54) PROCESSOR
(11) 62-262146 (A) (43) 14.11.1987 (19) JP
(21) Appl. No. 61-104619 (22) 9.5.1986
(71) HITACHI LTD (72) SHIGERU SHIBUKAWA
(51) Int. Cl⁴. G06F9/46,G06F9/42

PURPOSE: To execute the saving and returning of a register at high speed at the time of interruption and restarting of a processing, by providing a group of temporary registers having registers that can be saved and returned on the inside of a general-purpose processor MPU.

CONSTITUTION: Addresses indicated by a program counter and a status register PS3 are outputted from an address bus control circuit 13, and the content of the addresses is read out from a main memory by a data bus control circuit 12. An arithmetic control circuit 11 is operated by the instruction and the instruction is executed. A nest control register 4 controls the designation of the group of registers at the time of saving and returning operation of a group of temporary registers 9 and an internal register 1. Thereby, for the nest that can use the register group 9, saving and returning operation of the register 1 is made by using the register 1 without using the main memory, and by using the main memory only when the register group 9 is filled up.



2: control register

⑲日本国特許庁(JP)　⑪特許出願公開

⑫公開特許公報(A)　昭62-262146

⑤Int.Cl.⁴		識別記号	庁内整理番号
G 06 F	9/46	310	C-8120-5B
	9/42	330	7361-5B

⑬公開　昭和62年(1987)11月14日

審査請求　未請求　発明の数 1　(全7頁)

⑭発明の名称　処理装置

⑳特　願　昭61-104619

㉒出　願　昭61(1986)5月9日

㉗発明者　渋川　滋　勝田市市毛882番地　株式会社日立製作所那珂工場内

㉑出願人　株式会社日立製作所　東京都千代田区神田駿河台4丁目6番地

㉔代理人　弁理士　小川　勝男　外2名

明　細　書

1．発明の名称

処理装置

2．特許請求の範囲

1．処理装置内部に複数個の多目的汎用レジスタを使用した汎用処理装置において、割込みやサブルーチンコール等のプログラムの流れが変化し該処理実行後復帰する処理の場合、汎用処理装置内部に内部レジスタ全数コピー用のテンポラリレジスタおよびこのテンポラリレジスタ群と内部レジスタの退避、復帰動作時の該レジスタ群の指定を制御するネストコントロールレジスタの設置し、プログラム状態変化時に内部レジスタの退避、復帰の動作をテンポラリレジスタ群の使用が可能なネスト分は主記憶を使用することなく汎用処理装置の内部レジスタを使用し、テンポラリレジスタ群が満杯時のみ、主記憶を使用することを特徴とする処理装置。

3．発明の詳細な説明

〔産業上の利用分野〕

本発明は汎用処理装置（以下ＭＰＵと略称する）内部に複数の多目的レジスタを有する処理装置に係り、ＭＰＵ内部にそのレジスタを退避、復帰可能なテンポラリレジスタ群を設置し処理中断、再開時のレジスタ退避、復帰を高速に実行可能な処理装置に関する。

〔従来の技術〕

従来、この種の処理装置の典型的な例としてモトローラ(テ)社製、１６ビットＭＰＵ ＭＣ６８１０が知られている。本処理装置はデータレジスタを３２ビット×８、アドレスレジスタを３２ビット×８、その他コントロールレジスタを備え、データ，アドレスレジスタ合計で６４バイト分のレジスタが内蔵されている。第６図に当該処理装置も含め一般的な構成の例を示す。主記憶内に収納されている。プログラムを読出しその命令を実行し、次の命令を読出し……の様にプログラムに従つて処理を実行していくものである。第４図は、サブルーチンコールの動作を示したもので、１のルーチンを実行中に２により４よりの処理実行の要求が発生し

た場合、一般的には、２つの要求が発生（ＭＰＵで受付けられた時点）で次の実行する命令のアドレス３と、ＭＰＵ内のステータスレジスタを保存、一般的にはスタックポインタの示す主記憶に記憶し、４の先頭にジャンプし４よりの処理を実行する。４の処理の最後にはリターンサブルーチン等の命令となっている。この命令はスタックポインタの内容をＭＰＵ内部レジスタに戻す処理を行う。すなわち、次の実行する命令のアドレスここでは３、又２で実行していたステイタスレジスタの内容、これらがＰＣ（プログラムカウンタ）、ＳＲ（ステータスレジスタ）に戻され、３よりの実行が再開（離脱）される。

〔発明が解決しようとする問題点〕

最新のＭＰＵアーキテクチャは、処理するデータ長とＭＰＵ内部に多数の汎用レジスターを備えていることが、大きな特徴の一つである。

これは、高水準言語を効率よく、コンパイルタ、実行、最新のプログラミング技法を容易にするために実施されている。それは、多数の汎用レジスタをプログラムの変数等に使用することにより、ＬＳＩ、又はＰＭＵ内部設置レジスタの高速性の特長を生かして、演算性能向上に大きく貢献している。したがつて新しく開発される最新のＭＰＵの多くは、ますます内部汎用レジスタの本数増加データ長の増加の傾向にある。しかしこのＭＰＵ内レジスタ数の増加は処理の変化、例えばリアルタイム処理使用時のイベント応答、又はサブルーチンコール等のプログラムの流れが変化する（プログラムスイッチ）処理に対しては、デイメリットとして作用する。すなわち、再起動されたプログラムは、そのプログラム自身がＭＰＵ内の多数の汎用レジスタを最大限に利用して、プログラムを作成する。又高水準言語であれば、レジスタ類の使用はプログラムでまつたく意識する必要がないのが一般的である。したがつて、違うプログラムが起動される都に、基本的には、ＭＰＵ内部レジスタの全数を保存しておき、再実行時に前の状態に復帰する必要がある。このため、この退避、復帰の機能（前実行していたプログラムに戻れる）としてスタック機能が採用される。スタック機構とは、ＭＰＵ内部のスタックポインタを使用し記憶対象を主記憶とした、ＬＩＦＯ Last-In First-Out である。すなわちＰＣ(プログラムカウンタ)、ＳＲ（ステータスレジスタ）、各種コントロールレジスタ、汎用レジスタ類を、スタックポインタの示す主記憶アドレスに記憶する（これをプッシュと呼す）、該処理が実行後、状態を以上に戻すため、退避したデータ類を復帰する（これをポップと呼す）ことを実行している。この場合、プッシュ、ポップはＭＰＵ内部のデータを主記憶にライト又はリードの動作となり、主記憶の性能（アクセスタイム、サイクルタイム）が退避するデータ量に比例し、オーバヘッド時間として増加する。

外部イベントやサブルーチン等のプログラムを使用しない、処理内容であれば、高性能を示すが、実使用、特にリアルタイム処理の様にイベントの発生によるプログラムスイッチが頻繁に発生し、そのレスポンスタイムが問題となる分野においては、多数の汎用レジスタの設置が障害となる。

本発明の目的は多数の汎用レジスタを使用しても、プログラムスイッチ時等に処理性能を低下させないことにある。

〔問題点を解決するための手段〕

上記の目的はＭＰＵ内部の汎用レジスタ群を複数個設置とプログラムのネスティングをコントロールするネステコントロールレジスタを設置することにより達成される。

ネスティングとは第４図に示すように、プログラムの流れを中断し、必要とする処理を実行し、該処理の実行後、以前に実行していた処理に復帰し中断されていた処理を再開することで、１のルーチンより４のルーチンに実行の流れを変えることを１段のネスティングと一般的に称す。したがつて本発明は本来ならＭＰＵ内部に、汎用レジスタ群を無限数設置すれば、速度（プログラムスイッチ）に関して最大の効果が期待できる。これは不可能に近い。一般的なプログラムにおいて、割込処理のネスティング数はサブルーチンコール時のネスティング数よりも少ないと言われ、後者も

２～３段が一般的で、最大でも１０を越えることはない。したがつて２段程度のネスティングを高速に実行できれば処理全体を低下させることはない。

〔実施例〕

第１図に本発明の実施例を示す。

第１図はＭＰＵ内部のデータ処理部の構成を示したもので、前項より述べているように、１の内部レジスタが多数設置されていることが特長であり、又その語長が３２ビット、あるいは、６４ビット巾で、ＭＰＵの種類にもよるが、最大で、６４ケ位まで実装されているものまである。またこれからますます増加していく傾向にある。

２は、そのＭＰＵのアーキテクチヤに特徴な特殊なコトロールレジスタ等である。３は、命令の実行アドレスを示す。プログラムカウンタ、演算結果やモード等を表示、制御する、ステイタスレジスタである。一般的には、このプログラムカウンタとステイタスレジスタは、どのＭＰＵにも共通であり欠かかすことはできないものである。

ＭＰＵの動作は、３ＰＣ（プログラムカウンタ、命令のアドレスを示す）の示すアドレスを１３、アドレスバス制御回路より出力し、そのアドレスの内容を、１２データバス制御回路により主記憶より読み出す。その命令により１１の演算制御回路が動作し、命令を実行する。その他に演算制御回路は命令の読み出し、実行を含め、ＭＰＵ内のすべての動作を制御する。内部データの転送は、アドレス情報、演算データも含め１０のＭＰＵ内部バスを介して行なわれる。命令の実行は、１のＭＰＵ内部レジスタ間の演算もあるし、内部レジスタ外部主記憶、外部主記憶間もある。この演算結果は３：ステイタスレジスタに反映される。次にその命令が完了すると、ＰＣの内容が増加し、次の命令のアドレスを示し、その命令を主記憶より読み出す。このようにし、主記憶上の命令が次々と実行されていく。

この動作を第２図で説明すると１：通常処理は、命令フエツチがＰＣの内容を主記憶より読み出す、命令実行は、フエツチした命令を実行し、結果をステイタスレジスタ等に報告される。

例外処理はプログラム実行の流れが変化する場合を示す。すなわち、プログラムカウンタが次の命令のアドレス以外を示すことである。この場合１の通常処理内のイベント判定、検出が、プログラムの変化を示し、第４図の１のルーチンから、４のルーチンへのジヤンプ等となる。

ＭＰＵの動作は第４図の１のルーチンを実行時ＰＣは値を増加し次々と命令を順序よく実行する２のイベント例は、サブルーチンコールや例外処理を実行する場合、第１図の１１、演算制御回路がその結果により、３のＰＣの値を強制的に変化し必要ルーチンへ実行を移動する。この場合、第４図において１のルーチンでは、第１図の１、ＭＰＵ内部レジスタを無制限に使用しており、第４図、４のルーチンもまた、ＭＰＵ内部レジスタの使用を制限することは不可能である。したがつて、４のルーチンの実行前に１：のＭＰＵ内部レジスタを退避する必要がある。このＭＰＵ内部レジスタの退避に関して第１図の４ネストコントロールレジスタよりそのネスト回数（内部テンポラリレジスタの使用）を計数し、テンポラリレジスタ群の使用回数を制御する。

第１図、１のＭＰＵ内部ンジスタはすべてのレジスタが演算の対象として動作し、レジスタ⟷レジンスタ、レジスタ⟷主記憶の演算が可能であり結果がステイタスレジスタに反映される。

９：テンポラリレジスタ群とは、演算命令の対象として動作することはなく、ただの、レジスタを示すすなわち、ＭＰＵ内部レジスタのコピーとしてデータを一時記憶するために使用する、リード、ライト可能レジスタである。次に第５図と第２図、サブルーチンコール又は割込み時のＭＰＵ内動作、第３図に復帰（リターン命令）の動作を使用し、プログラムのネスティングの動作を示す。第５図の１のルーチンをプログラムが実行していると、２のイベントにより、４のプログラムへの処理変更要求が発生する。この場合処理装置は４にジヤンプするための準備を実行する。すなわち、３に戻る処理、３のアドレスを示すプログラムカ

ヤタのコピーと、2のイベント印加時点の演算結果収納レジスタ、ステータスレジスタのコピーを作成する。一般的なMPUであればこの場合、イベント又はサブルーチンコール命令の実行により該当ジャンプ先アドレスを計算し、前記の2つのレジスタの内容をスタックレジスタの示す主記憶に格納し、スタックポインタを収納アドレス分移動し、第5図の4の先頭アドレスにジャンプする。しかし、この場合、4の先頭アドレスよりの命令の実行について、第1図に示す。1のMPU内部レジスタの退避は行なわれておらず、第5図の4の先頭よりのルーチンで内部レジスタを使用するのであれば、ルーチン1で使用していた、レジスタ類の保存を行なわなければならない。この様に内部に複数の汎用レジスタを、多数有するMPUは、一般的に、この内部レジスタの、退避、復帰の命令を有している。すなわちそれらの手法は、スタックポインタの示す主記憶上のアドレスに、内部レジスタの全数、あるいは指定するレジスタを以前に退避した、PC，SRの次に、退避するものである。当然、多数の内部レジスタがあつて、そのプログラムの作成を高水準言語で行なえば、内部レジスタの使用等を、プログラムは、知ることも出来ない。したがつて、プログラムの流れが変化するたびに、内部レジスタは全数退避しないと、完全な処理は実行不可能となる。

このことは内部レジスタを、64ケ、32ビット巾を有するMPUを例にとつてみれば、システムバス巾を16ビットの場合、64×2＝128ワード（1ワード＝16ビット）のデータの転送を意味し、仮に1ワード、1μs（10⁻⁶SEC）の動作の主記憶であれば、128μs×2＝256μs（退避と復帰）が最低必ずオーバーヘッドとして発生する。すなわち、内部レジスタの退避は必ず主記憶の動作が伴い、これがオーバーヘッド時間の大きな要因である。

第5図によれば、4，6，8にて128μsの退避動作、9，7，5にて128μsの復帰動作が必要となる。もちろんSR，PCの退避、復帰時間もこの時間に加算される。

本発明は第1図に示すように、9のテンポラリレジスタ群、1：内部汎用レジスタのコピーと2と4、ネストコントロールレジスタを設置することによる。この方式によれば、第5図について、1のルーチン実行中に2の4に変化のイベントが発生すると、第2図にお序論、実行中の命令の終了直後、イベントの有無検出（サブルーチンコールを広義のイベントととする）でイベンドであれば、PC，SRをコピー、スタッキングする。次にネストコントロールレジスタの内容を調べる。ネストコントロールレジスタはMPU内部に設置した汎用レジスタのコピー（テンポラリレジスタ）の指示を制御を行うもので、2のイベント時、MPU内部のテンポラリレジスタは未使用で"0"となつている。ここでは当然、第5図の1のルーチンで使用していた、汎用レジスタのすべてが、第1図、5のテンポラリレジスタ（コピー1）に転送される。この動作は主記憶の動作を必要とせずMPU内部で実行可能であり、MPU動作クロックに近い速度が終了が可能である。仮に10MHzのクロックで動作していたとして1クロックでレジスタの転送が実行可能として1クロック100μsで64ケのレジスタは64×0.1μs＝6.4μs で転送が終了する、このことは単純に計算しても主記憶を使用した場合の約20倍の速度となる。

第2図のテンポラリレジスタコピーが終了するとネストコントロールレジスタの値を＋1し、イベントアドレス計算を実行し、第5図の4にジャンプする。すると4よりの実行ルーチンでは、汎用レジスタの内容すべてが退避されているため、内部レジスタのすべてが使用可能となる。

次に第5図において4のルーチンより6のルーチンに変更要求が発生すると、2の動作と同じ動作が実行される。この場合MPU内部のテンポラリレジスタ（この場合2）が1つ空であり、第2図の内部レジスタコピーテンポラリレジスタコピーが実行されネストコントロールレジスタが＋1される。次に第5図において6のルーチンより8のルーチンに再度変更要求が発生すると、今回は

ＭＰＵ内部のテンポラリレジスタが２ケともデータが退避されており、内部レジスタの内容はスタックポインタの示す主記憶に退避される。このようにして、２段のネスティングまでは通常の主記憶を使用した退避法の約数１０倍の速度で処理が実行される。

次に復帰の場合は第５図において８のルーチンの実行が終了し９においてリターン命令が実行される。リターン命令は一般的にサブルーチンコール命令等と対をなす命令で、サブルーチンコール命令でＳＲ，ＰＣをスタックに退避したならば、そのＳＲとＰＣをＭＰＵ内部に戻し、スタックレジスタのアドレスを変更し退避前のアドレスに変し、退避前の処理に戻すものである。

この場合はＳＲ，ＰＣの復帰に伴う多量の内部レジスタの復帰がある。すなわち、第５図の９からの復帰はＭＰＵ内部テンポラリレジスタが満杯であり、主記憶に退避されている。

これを第３図のリターン命令の例で示す。本命令が実行されると最初にネストコントロールレジスタの内容を調べ、２以上であれば、ＭＰＵ内部のテンポラリレジスタが満杯であり、スタックポインタの示す、主記憶に汎用レジスタの内容が退避されていることを示す。このため動作は、スタックポインタの示す主記憶よりＭＰＵ内部レジスタにデータを復帰し、ネストレジスタの値を－１する。次にＳＲとＰＣを主記憶より復帰しＭＰＵ内部のレジスタに復帰することにより、第５図の６のルーチンに戻る。６のルーチンの最後の７にてまたリターン命令が実行される。

今回のリターン命令は、ネストコントロールレジスタの値が２未満であり、退避されたレジスタデータはＭＰＵ内部のテンポラリレジスタに収納されていることを示している。したがつて動作は、退避と同様にＭＰＵ内部で高速に復帰が実行され第５図の４のルーチン６にジャンプした次の命令より実行が再開される。このようにして、５のリターン命令で、１のルーチンの３に復帰するようにネスティングの逆動作が実行されていく。

【発明の効果】

本発明によれば、プログラム変更時のネスティング２段まではＭＰＵ内部に多数の汎用レジスタを有する。処理装置において、主記憶の動作速度とＭＰＵ内部動作の差分、現在の技術で約１０倍～数１０倍（今後ますます大きくなる）の処理速度の改善効果がある。又ネストレジスタの値、スタックポインタの値、ＭＰＵ内部テンポラリレジスタのアドレッシングを考慮することにより、プログラムより、退避したレジスタの内のデータの加工等処理が可能でもある。

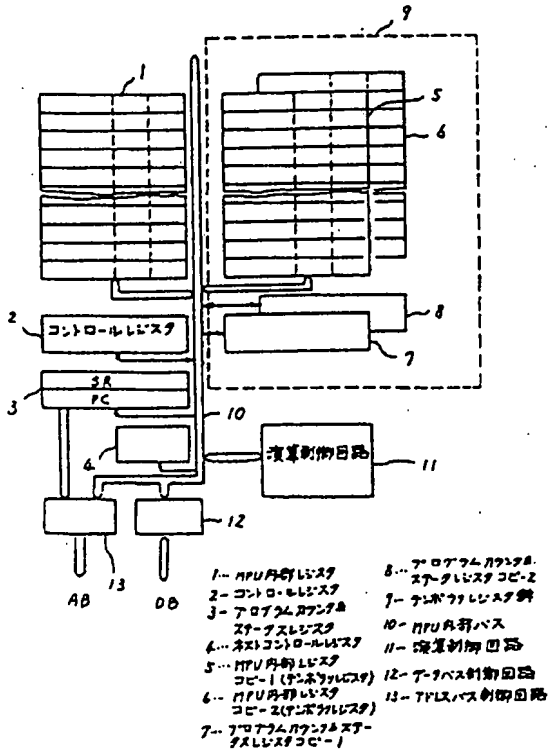
すなわちＭＰＵ内に汎用レジスタコピー用テンポラリレジスタ部とテンポラリレジスタをコントロールするネストコントロールレジスタ、設置により、プログラムスイッチ動作によるオーバーヘッドを大巾に低減することが可能となり処理装置全体の性能を大巾に向上することが可能となる。

４．図面の簡単な説明

第１図に本発明の実施例の構成図を示す、第２図に本発明のプログラムスイッチ時の動作フローの概略を示す、第３図に本発明のプログラムスイッチ時よりの復帰のリターン命令の概略フローを示す、第４図に一般的なプログラムスイッチの動作を示す、第５図に本発明のネスティング動作とレジスタ退避動作説明図を示す、第６図に一般的な処理装置の概略構成図を示す。

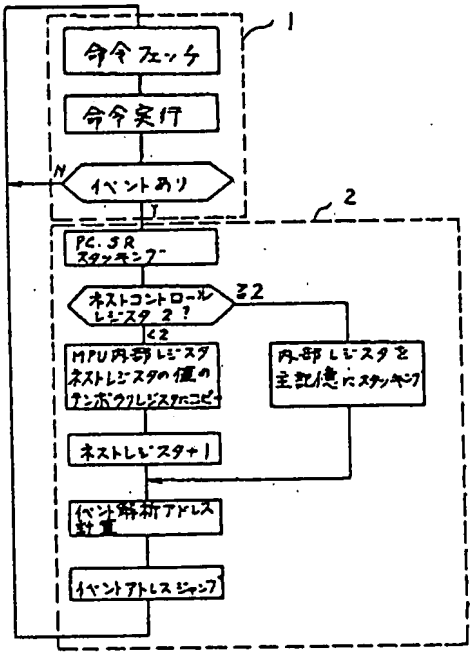
代理人　弁理士　小川勝男

第 1 図

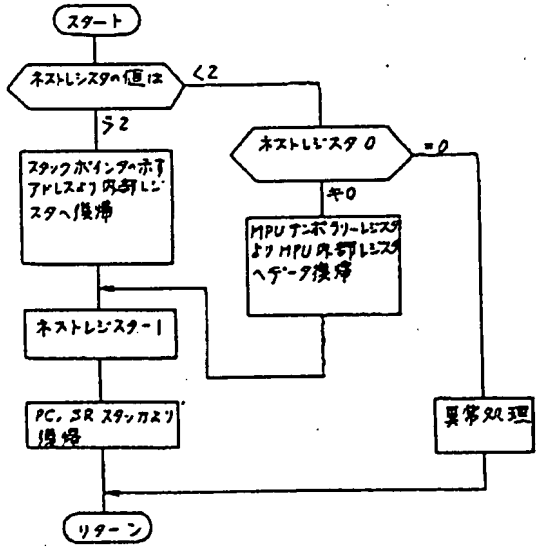


1… MPU内部レジスタ
2… コントロールレジスタ
3… アログラムカウンタ&ステータスレジスタ
4… ネストコントロールレジスタ
5… MPU内部レジスタコピー1(テンポラリレジスタ)
6… MPU内部レジスタコピー2(テンポラリレジスタ)
7… アログラムカウンタ&ステータスレジスタコピー1
8… アログラムカウンタ&ステータスレジスタコピー2
9… テンポラリレジスタ群
10… MPU内部バス
11… 演算制御回路
12… データバス制御回路
13… アドレスバス制御回路

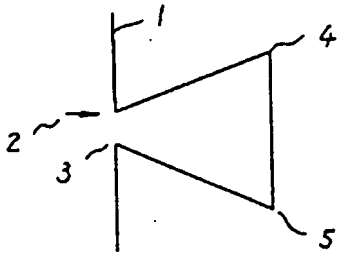
第 2 図



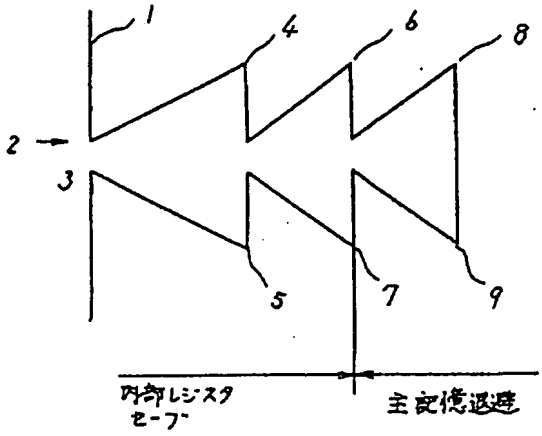
第 3 図



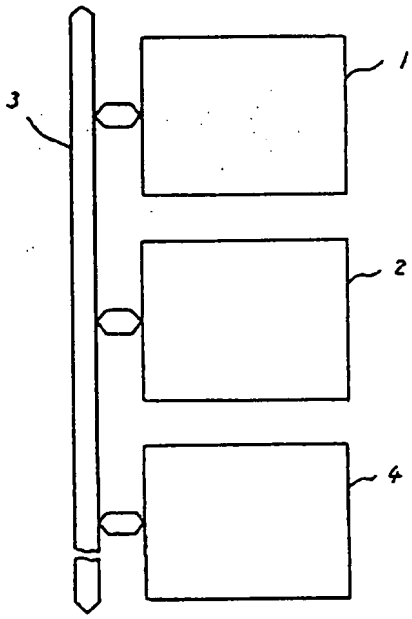
第 4 図



第 5 図



第 6 図



Partial translation of JP-A-62-262146

"An embodiment of the present invention is shown in Fig. 1.

Fig. 1 shows a construction of a data processing portion in MPU which is characterized in that a plurality of internal registers identified by 1 are arranged as set forth in the preceding section, a word length of which is 32 bits or 64 bits. While variable depending upon a type of MPU, there are some data processing portion, on which sixty-four internal registers are mounted at the maximum. Number of the internal registers to be mounted tends to be increased progressively.

Numeral 2 denotes a special control register or the like characterizing an architecture of MPU. Numeral 3 denotes a program counter representative of execution address of instruction and a status register displaying and controlling result of calculation, mode and so forth. In general, the program counter and the status register are common for any MPU and inevitable.

Operation of MPU is to output an address represented by the PC 3 (program counter, indicative of the address of instruction) from an address bus control circuit 13, and the content of the address is read out from a main memory by a data

bus control circuit 12. By the instruction, a operation control circuit 11 operates to execute the instruction. In addition, the operation control circuit reads out the instruction and controls all of operation in MPU including execution of instruction. Transfer of the internal data including address information, calculation data is performed via internal bus 10 of MPU. Execution of instruction includes operation between internal registers of MPU 1, operation between an internal register and the external main memory, and within the external main memory. The result of operation is reflected in the status register 3. Next, when an instruction is completed, the content of the PC is increased to indicate an address of next instruction to read out the instruction from the main memory. Thus, instructions on the main memory are executed sequentially.

Discussing the operation with reference to Fig. 2, in normal process 1, the content of the PC is read out from the main memory in instruction fetch. The fetched instruction is executed in instruction execution. The result is reported to the status register or the like.

Exception process represents the case where flow of

program execution is varied. Namely, the program counter indicates an address other than the address of next instruction. In this case, event judgment and detection in the normal process 1 indicates variation of the program, such as jumping from a routine 1 of Fig. 4 to a routine 4.

Operation of MPU is to sequentially execute instructions by increasing the value of the PC during execution of the routine 1 of Fig. 4. An example of event 2 is that upon sub-routine call or execution of exception process, the operation control circuit 11 of Fig. 1 forcedly varies the value of the PC 3 to shift execution to necessary routine. In this case, in the routine 1 of Fig. 4, the internal registers of MPU 1 are used in an unrestricted manner. Also, routine 4 of Fig. 4 is also not possible to restrict use of the internal registers of MPU. Accordingly, before execution of the routine 4, it becomes necessary to save the internal registers of MPU 1. Concerning saving of the internal registers of MPU, number of times of nesting (use of internal temporary register) from a nest control register 4 of Fig. 1 is counted to control number of times of use of the temporary register portion.

The internal register of MPU 1 of Fig. 1 operates with

taking all of registers as objective for calculation and is capable of operation between registers, between a register and the main memory. The result is reflected in the status register.

Temporary registers 9 do not operate with taking operation instruction objective, and are registers which can be read and write, to be used for temporarily store data as simple registers, namely a copy of the internal registers of MPU. Next, operation of MPU upon sub-routine call or interrupt is shown in Figs. 5 and 2, operation for restoring (return instruction) is used for operation of nesting of the program is shown in Fig. 3.

When the program executes a routine 1 of Fig. 5, a demand for process modification to the program 4 occurs. In this case, the processing apparatus executes preparation for jumping to 4. Namely, a process 3 to return to produce a copy of the program counter 3 indicative of the address and a copy of the operation result storage register upon application of event and the status register. In case of general MPU, an address of destination to jump is calculated by execution of the event or sub-routine call instruction, the contents of two registers

are stored in the main memory indicated by a stack register, a stack pointer is shifted by an amount corresponding to the stored addresses to jump to the leading address 4 of Fig. 5. However, in this case, execution of instruction from the leading address 4 is shown in Fig. 1. Saving of the internal registers in MPU 1 is not performed. If the internal registers are to be used from the routine of the leading address 4 of Fig. 5, saving of the registers used by the routine 1 has to be performed. As set forth above, MPU having a large number of general purpose registers therein has instructions for saving and restoring of the internal registers, in general. Namely, those methods are to save all of the internal registers or the designated registers next to the preliminarily saved PC and SR in the address on the main memory indicated by the stack pointer. Naturally, when a large number of internal registers are present and preparation of the program is performed by a high-level language, use of the internal registers or the like cannot be known by the program. Accordingly, every time of variation of flow of the program, complete execution becomes impossible unless all of the internal registers are saved."

"4. BRIEF DESCRIPTION OF THE DRAWING

Fig. 1 shows an illustration of a construction of an embodiment of the present invention;

Fig. 2 shows a general operation flow upon switching of program of the present invention;

Fig. 3 shows a general flow of a return instruction upon returning upon switching of the program of the present invention;

Fig. 4 shows operation of general program switching;

Fig. 5 is an explanatory illustration of nesting operation and register saving operation of the present invention; and

Fig. 6 is an illustration showing general construction of general processing device."

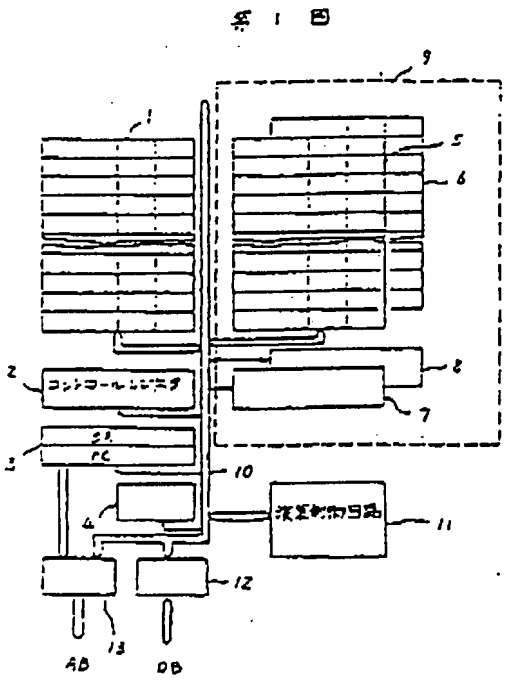


Fig. 1

1 INTERNAL REGISTERS OF MPU

2 CONTROL REGISTER

3 PROGRAM COUNTER AND STATUS REGISTER

4 NEST CONTROL REGISTER

5 COPY 1 (TEMPORARY REGISTER) OF INTERNAL REGISTERS OF MPU

6 COPY 2 (TEMPORARY REGISTER) OF INTERNAL REGISTERS OF MPU

7 COPY 1 OF PROGRAM COUNTER AND STATUS REGISTER

8 COPY 2 OF PROGRAM COUNTER AND STATUS REGISTER

9 TEMPORARY REGISTERS

10 INTERNAL BUS OF MPU

11 OPERATION CONTROL CIRCUIT

12 DATA BUS CONTROL CIRCUIT

13 ADDRESS BUS CONTROL CIRCUIT

FIG. 2

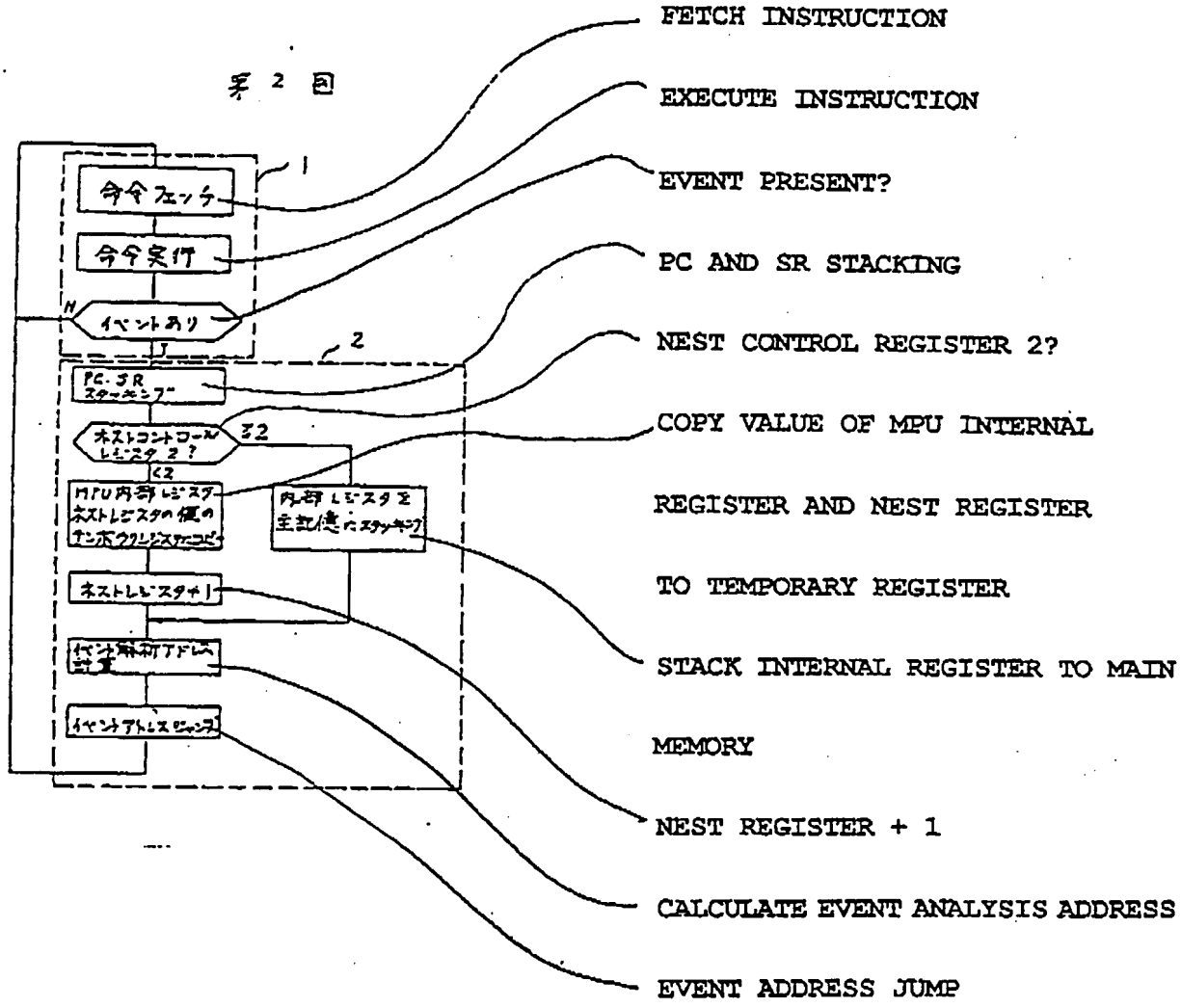
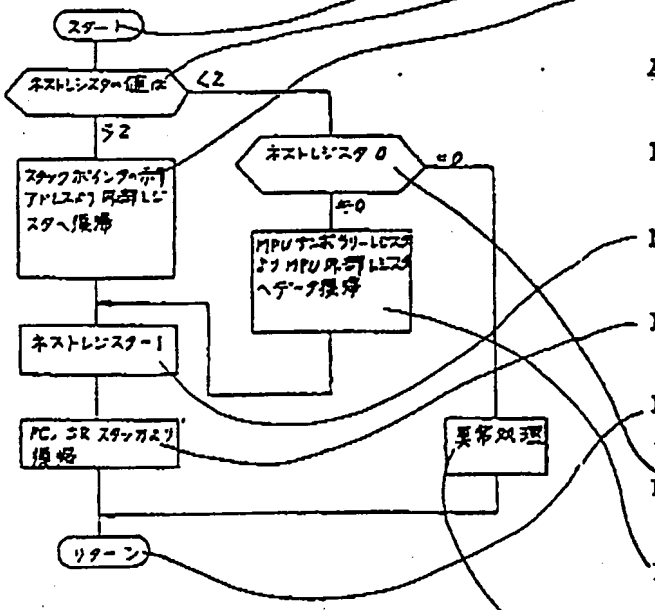

FETCH INSTRUCTION
EXECUTE INSTRUCTION
EVENT PRESENT?
PC AND SR STACKING
NEST CONTROL REGISTER 2?
COPY VALUE OF MPU INTERNAL
REGISTER AND NEST REGISTER
TO TEMPORARY REGISTER
STACK INTERNAL REGISTER TO MAIN
MEMORY
NEST REGISTER + 1
CALCULATE EVENT ANALYSIS ADDRESS
EVENT ADDRESS JUMP

FIG. 3



START

VALUE OF NEST REGISTER?

RESTORE INTERNAL REGISTER FROM ADDRESS INDICATED BY STACK POINTER

NEST REGISTER -1

RESTORE PC AND SR FROM STACK

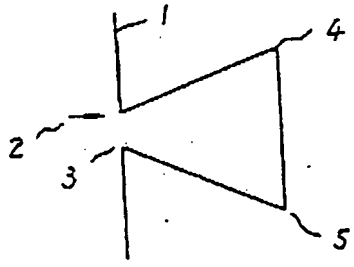
RETURN

NEST REGISTER 0?

RESTORE DATA FROM MPU TEMPORARY REGISTER TO MPU INTERNAL REGISTER

ABNORMAL PROCESS

第 4 図



第 5 図

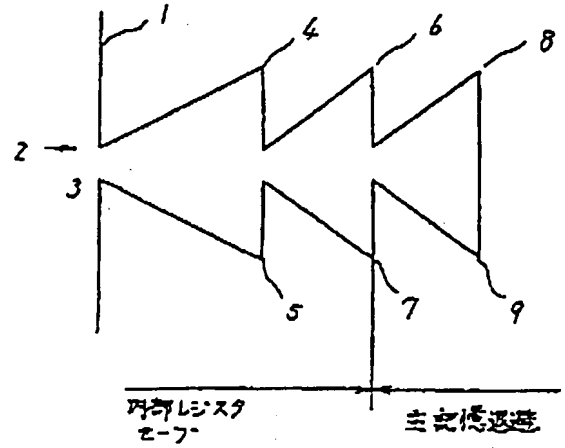


FIG. 5

SAVE INTERNAL REGISTERS

SAVE MAIN MEMORY

第 6 図